MUSS

USER MANUAL

VOLUME 3

UNIVERSITY OF MANCHESTER ISSUE 12

DRAFT: October 2019

The MUSS User Manuals were published in three volumes:

```
MUSS   USER    MANUAL VOLUME 1:  Basic Facilities of MUSS

MUSS LANGUAGE  MANUAL VOLUME 2:  High-level Languages

MUSS UTILITIES MANUAL VOLUME 3:  Utilities
```

It may be helpful to read this Manual in conjunction with the MUSS User Manual Vol. 1 Issue 12, 1986 which can be found at archive.org .

This edition of Volume 3 was intended to be for Issue 12. However, it is an undated draft with several parts missing or incomplete. I have not found a more complete version.

This text was created by OCR conversion of the original, followed by extensive editing. For the most part the OCR was quite good. However, since in the original text "." and "," were very similar, this was a common error.

There was also a problem with use of quotes. I have not yet decided how best to handle this, so there is some inconsistency in the text.

Few typos were found in the original text and none were serious. Just a few "off by one" index errors, most quite obvious. I left the spelling checker set to UK English.

Any comments or errors found should be reported to me at:

michaeldgodfrey@gmail.com.

October 9, 2019

INTRODUCTION

This Manual describes the facilities which are available as an adjunct to the basic MUSS system described in the MUSS User Manual. These utilities may not be available on all machines, although the source for them can be issued to those installations requiring them. A description of how to compile the utilities is given in the *Appendix Implementation Manual* for each machine. From a user point of view the utilities generally exist as libraries, usually in the directory UTIL::. However the method of access to each utility is given with its description in this Manual, although this may change at any particular installation.

Contents

USER MANUAL i

DRAFT: October 2019 iii

INTRODUCTION v

Contents vi

1 EDITORS 1
- 1.1 EMACS EDITOR . . . 1
  - 1.1.1 Introduction . . . 1
  - 1.1.2 How to access the EMACS editor . . . 1
  - 1.1.3 Form of Commands . . . 1
- 1.2 SUMMARY OF COMMANDS . . . 2

2 MUNX 7
- 2.1 INTRODUCTION . . . 7
  - 2.1.1 How to access MUNX . . . 7
- 2.2 HIGH LEVEL COMMANDS . . . 8
- 2.3 SYSTEM CALL PROCEDURES . . . 11
  - 2.3.1 File Control Procedures . . . 11
  - 2.3.2 Tasks, Signals and Interrupts Procedures . . . 12
  - 2.3.3 Miscellaneous Procedures . . . 13
  - 2.3.4 String Utility Procedures . . . 13
  - 2.3.5 Mathematical Procedures . . . 14
  - 2.3.6 Renamed MUSS Functions . . . 14
  - 2.3.7 High level I/O Procedures . . . 15
  - 2.3.8 Macro Procedures . . . 16
  - 2.3.9 Extensions . . . 16

3 THE CORE GRAPHICS PACKAGE 17
3.1 CORE GRAPHICS FACILITIES 17
3.1.1 Introduction 17
3.1.2 How to access CORE 17
3.1.3 Core System Overview 18
3.1.4 Commands 21
3.2 GKS GPAPHICS 37
3.3 DIRECT ACCESS GRAPHiCS 37
3.3.1 ATV Graphics 37

4 TEXT WINDOW FACILITIES 41
4.1 TEXT WINDOW FACILITIES 41
4.1.1 Introduction 41
4.1.2 How to Access the Text Window Facilities 41
4.1.3 Commands 41

Index – Volume 3 44

1 EDITORS

A basic editor and a screen editor are provided with the basic facilities of MUSS. This Chapter describes alternative editors which are available for use under MUSS.

1.1 EMACS EDITOR

1) EMACS (ADDR [LOGICAL8], LOGICAL64)

This is an 'EMACS'-type editor. The parameters given are P1 a filename for the file to be edited and P2 a terminal type.

1.1.1 Introduction

EMACS is a screen editor which is widely available, although not all of its large set of commands are implemented in each version. This section describes the commands available in the MUSS implementation.

The screen display is divided into two main areas. The largest area at the top of the screen displays the text being edited, the cursor will normally appear at the 'point' where editing is taking place, (the point (see later) is considered to be immediately before the character at which it is pointing), long lines are wrapped on to succeeding lines and are terminated with a '!'. The bottom two lines on the screen are used to display a line of monitoring information about the mode in which the editor is working. The files or buffer currently being edited and a line on which arguments of commands are displayed and/or typed. In addition, any error messages are displayed in this area.

1.1.2 How to access the EMACS editor

The editor exists as a library in the directory of user UTIL and is made available by opening this library with the command LIBRARY UTIL:EMACS.

1.1.3 Form of Commands

The method of entering commands is designed to allow the appropriate action to be taken with the minimum of keystrokes.

1.2 SUMMARY OF COMMANDS

1) Control Commands

C – implies Control Key depressed (with alphabetic characters upper or lower case have the same effect).

C-@	Set mark at current position
C-a	Move to start of current line
C-b	Move back one character
C-d	Delete character
C-e	Move to end of current line
C-f	Move forward one character
C-h	Move back one character
C-i	Insert tabulator
C-j	Insert newline
C-k	Delete text to end of current line and push onto kill buffer
C-l	Redisplay screen
C-m	Insert newline
C-n	Move down one line (staying in same column)
C-o	Insert newline and back a character
C-p	Move up one line (staying in same column)
C-q	Insert new character unchanged
C-r	Start reverse search for string
C-s	Start forward search for string
C-t	Interchange characters at point and move forward one character afterwards
C-u	Form argument
C-v	Move forward a screen
C-w	Delete text between point and mark and push onto kill buffer
C-x	Control-x commands prefix
C-y	Copy from kill buffer to point, set mark at start of inserted text and point at end (‘Yank kill buffer’)
C-z	Exit from editor
C-ESC	Meta commands
C-DEL	Delete character
C-_	No effect. (Null command)

2) Meta Commands

M - Implies command is preceded by ESC or C-ESC. For alphabetic commands upper and lower case have the same effect.

M-a	Move to start of current sentence
M-b	Move backwards one word
M-c	Capitalise the following word
M-e	Move to end of current sentence
M-f	Move forwards one word
M-g	Fill text in region
M-h	Move to start of current paragraph and place mark at end of paragraph
M-k	Delete following word and push onto kill buffer
M-l	Change following word to lower case
M-o	ANSI function key sequence (see later)
M-s	Centre current line
M-u	Change following word to upper case
M-v	Move backwards a screen
M-w	Push a copy of the region onto the kill buffer
M-x	Execute a MM command
M-y	(Use only after C-y or previous M-y). Replaces text reinserted by previous command (between point and mark) with previous kill buffer i.e. next down kill chain
M-<	Move to start of buffer
M->	Move to end of buffer
M-=	Count lines in region
M-;	Insert comment
M-[	ANSI cursor control functions
M-DEL	Delete previous word and push onto kill buffer
M-C-[	Move to start of paragraph
M-]	or
M-C-]	Move to end of paragraph
M-0→M-9	Form arguments.

3) Control-X Commands

This set of commands is preceded by a C-x character. If the command following is alphabetic upper or lower case may be used.

C-x a	The next delete is to use the current kill buffer
C-x b	Switch buffer
C-x e	Execute keyboard macro
C-x f	Set right-hand margin for fill
C-x h	Set region as entire buffer
C-x k	Kill current buffer
C-x >	Set mark at end of buffer
C-x <	Set mark at start of buffer
C-x (	Start keyboard macro
C-x)	End keyboard macro
C-x .	Set fill prefix
C-x C-b	list buffers
C-x C-f	Find file and read into buffer of same name
C-x C-l	Change region into lower case
C-x C-s	Save current buffer into its associated file
C-x C-u	Change region into upper case
C-x C-v	Visit file - read file into buffer
C-x C-w	Write out buffer to file after prompting for file name
C-x C-x	Exchange point and mark
C-x C-z	Exit from editor.

4) MM Commands

These are preceded by the sequence M-x. The command name is typed into the prompt area and any arguments given are enclosed in escape characters.

The commands are:

Adelimiters
Autofill
Autosave
Include
Overwrite
Query Replace
Rdelimiters
Replace
RReplace
Untabify

5) Function Keys

6) Keypad

On terminals which transmit an ANSI alternate sequence from the keypad then they may be used to generate arguments to other commands.

2 MUNX

2.1 INTRODUCTION

MUSS provides facilities to enable the user to generate his own UNIX[1] style environment. The UNX supervisor creates a MUSS process that executes a shell. A 'C' compiler, a command preprocessor, a standard 'C' IO interface, and Unix system call and library procedures are provided. Using these the user can compile 'C' programs to create a complete environment.

There are some deviations and restrictions in system call procedures.

The major difference concerns the organisation of filestore and access rights. The filestore consists of user directories, with a hierarchy of subdirectories stemming from each user directory. The file directories and subdirectories are not accessible to user programs. In a pathname user directories are identified by a following ':', e.g. @STUD:MHK/LAB.PROGS/SORT.C. While in the change directory command 'ch.dir' the user's password is needed when changing between user directories, e.g. @STUD:X:MHK/LAB.PROGS where STUD is the username and X his password. In a pathname list '@' identifies the start of a pathname. Multiple directory entries for the same file, achieved by the system procedure link, is not available. File access permissions are granted on files to users. Access rights are only verified when the user directory name appears in a file pathname. There is no equivalent to Unix 'group' and 'all users' access rights. Furthermore, there is no setting of the user id to the effective user id associated with executing a program. The object programs of available Unix-like commands are in BINX:

The following terminal control characters are not yet Implemented; '@' (erase line), '#' (send kill signal), ASCII FS (send quit signal).

2.1.1 How to access MUNX

Normal access to MUNX is by way of the UNX supervisor (c.f. JOB supervisor for normal MUSS usage).

[1] Trademark of Bell Labs.

2.2 HIGH LEVEL COMMANDS

1) cc (1)

NAME

cc – MUSS C Compiler (see Volume 2. Chapter 5).

pcc – Portable C compiler.

SYNOPSIS

cc [option] ... file ...

pcc [option] ... file ...

DESCRIPTION

cc is the MUSS C Compiler.

File arguments whose names end in '.c' are taken to be C source programs. The output from the compilation will be the file 'a.out', unless the '-o' option is given (see below).

The following options are interpreted by cc.

– p	Run only the macro preprocessor, and place the result for each '.c' fife in a corresponding '.i' file.
– c	Do not invoke the macro-preprocessor, and immediately compile the '.c' files, which will be assumed to be in a preprocessed format.
– Dname = def	
– Dname	Define the <name> to the preprocessor, as if by '#define'. If no definition is specified, the name is defined as 1.
– Uname	Remove any initial definition of <name>.
– o output	Name the final output file <output>.
– z size	Specify the size of the Run Time Machine for the programs being complied, as follows

```
0 As compile time machine
1 16 Bit ADDR 16 Bit INTEGER
2 32 Bit ADDR 32 Bit INTEGER
3 32 Bit ADDA 16 Bit INTEGER
```

If this option is not specified the default value is 0.

NAME

pcc – Portable C complier.

Specification similar to Unix portable C compiler.

2) cpp (1)

NAME

cpp – Macro pre-processor for 'C' compiler.

SYNOPSIS

cpp [optlon] . . . [flle].

DESCRIPTION

cpp is the macro-preprocessor used by the MUSS and portable C compilers. Unless at least one file is specified, the preprocessor uses the standard input. Output is sent to the standard output unless a second file is specified.

The following options are interpreted by the pre-processor.

– p	Do not produce line number references in the output file.
– c	Do not copy comments to the output file.
– Dname = def	
– Dname	Defines the name as if by '#define'. If no definition is given, the name is defined as 1.
– Uname	Remove any initial definition of name.
– I dir	Search for files in 'dir' before following standard search pattern.

3) ed (1)

NAME

ed – MUSS editor EDIT (see Volume 1, Chapter 5) .

SYNOPSIS

ed input.file output.file

DESCRIPTION

Both file arguments must be provided. For a description of editor commands see the MUSS User Manual Volume 1, Chapter 5.

4) emacs (1)

NAME

emacs – MUSS EMACS editor (see Chapter 1 of this Manual).

SYNOPSIS

emacs –termtype filename

DESCRIPTION

The termtype option gives the terminal type on which the editor is to be used. Currently available options are UTS10, NBURY (Newbury 7003), ITUBE (intertube), ZENITH (Zenith Z-29), VT100 or ANSI. If no option is given ANSI is assumed. The second parameter gives the filename of the file on which editing will commence. If no filename is given an empty buffer is created.

5) sh (1)

NAME

sh – command language.

SYNOPSIS

sh [–ceiknrstuvx] [arg] ...

DESCRIPTION

sh is a command programming language that executes commands read from a terminal or a file.

For a complete description of this command, see the 'UNIX Programmer's Manual. Seventh Edition'.

However, in addition to the built-in commands a facility is provided to enable calls on MUSS library procedures to be made directly, without causing a fork, which would be unnecessary and inefficient. The format for this facility requires the symbols '**' preceding the MUSS library command to be executed, e.g.,

**CAPTION HELLO!20!WORLD!OA!

Before invoking the command the usual shell argument substitutions will be performed (except on the initial '*' symbols).

This mechanism is particularly useful for opening MUSS libraries as otherwise in the Unix environment they would be opened in a sub-process and closed as that sub-process ended.

If the MUSS command contains shell meta characters then they should be quoted using the " " mechanism.

For usage see the description of LIB[RARY] in the MUSS User Manual Volume 1.

2.3 SYSTEM CALL PROCEDURES

A subset of the Unix system call procedures is available under MUNX. They are intended to emulate commands described in the 'Unix Programmer's Manual, Seventh Edition'. For clarity, a full specification of the commands is omitted here. Only deviations and worthy points of note are documented for these procedures.

2.3.1 File Control Procedures

```
CREAT  (NAME, MODE) opens a new output file.
OPEN   (NAME, MODE) opens an existing file.
DUP    (fd) duplicates a reference to an open file.
DUP2   (fd1, fd2) makes the file descriptor fd2 refer to the
        open file fd1.
PIPE   (fd [2]) sets up a pipe and returns 2 file descriptors,
        fd [0] for the read end of the pipe and fd [1] for
        the write end.
```

```
CLOSE  (fd) close the file/pipe fd.
CHDIR  (dirname) change default directory.  If dirname starts
        with '/name/' then directory name given is relative to
        current user directory.
        If dirname starts with '/username:password:'  then user
        directory is changed.  For blank passwords 'password:'
        may be omitted.
LINK   (fd) add a second reference to an existing file.  (This is
        not implemented.)
CHROOT (dirname) not implemented.
UNLINK (fd) delete a file reference.  Since multiple references
        to a file are not possible in MUSS, this will delete the
        file.
READ   (fd, buffer, nbytes) read from file fd up to nbytes.
WRITE  (fd, buffer, nbytes) write nbytes to file fd.
LSEEK  (fd, offset, direction) move to a new position within a file.
TELL   (fd) return the current position in the file.
STAT   (name, buff) return details of file name in buff (Not yet
        implemented).
FSTAT  (fd , buff) return details of open file fd in buff.
ISATTY (fd) checks if a file is connected to a terminal.
```

2.3.2 Tasks, Signals and Interrupts Procedures

```
FORK   () create a new task with the same environment as the caller.
EXECL  (name, arg0, arg1, ..., argn, 0) overlay the current program
          with name.
EXECLE (name, arg0, arg1, ..., argn, 0, env) overlay the current
          program with name and pass environment env.
EXECV  (name, argv) overlay the current program with name.
EXECVE (name, argv, env) overlay the current program with name
         and pass environment env.
XEXIT  (status) terminate this task and pass status to parent.
WAIT   (status) wait for a child to terminate and receive status.
GETPID (pid) get this task's unique pid number.
NICE   (incr) reduce task priority (Not yet implemented).
TIMES  (buffer) return some statistics about this task and its
         children.
ALARM  (seconds) go to sleep for a given time.
KILL   (pid, sig) send an interrupt to a task.
```

```
PAUSE  () wait for an interrupt.
SIGNAL (sig, func) change the interrupt handler for interrupt sig.
```

2.3.3 Miscellaneous Procedures

```
BRK     (adr) set the amount of dynamic memory.
SBRK    (incr) set more dynamic memory.
TIME    (buff) get the time and date.
STIME   (buff) set the time and date.
IOCTL   (fd, req, struc) report/change file or terminal options.
STTY    (fd, struc) change terminal options.
GTTY    (fd, struc) report terminal options.
SETJMP  (env) save environment for non-local jump.
LONGJMP (env, value) make a non-local jump.
```

2.3.4 String Utility Procedures

IS . . . check for character in class

```
   ISALNUM is character alphanumeric.
   ISALPHA is character alphabetic.
   ISASCII is character valid ASCII.
   ISCNTL  is character a control character.
   ISDIGIT is character a digit.
   ISLOWER is character lower case.
   ISPRINT is character a printable character.
   ISPUNCT is character a punctuation character.
   ISSPACE is character a separator character.
   ISUPPER is character upper case.
   ISXDIGIT is character a hexadecimal digit.
```

```
STRCAT  (str1, str2) add str1 to str2.
STRNCAT (str1, str2, max) add at most max char from str1 to str2.
STRCMP  (str1, str2) compare str1 and str2.
STRNCMP (str1, str2, max) compare str1 and str2 up to max
       characters.
STRCPY  (dest, source) copy source to destination.
STRNCPY (dest, source, max) copy source to destination up to max
       characters.
STRLEN (str) return length of string.
INDEX  (str, char) find first character char in string.
RINDEX (str, char) find last character char in string.
```

```
TOLOWER (char) convert character to lower case.
TOUPPER (char) convert character to upper case.
```

2.3.5 Mathematical Procedures

```
EXP   (x) return the exponent of x.
LOG   (x) return the natural log of x.
LOG10 (x) return the base 10 log of x.
POW   (x, y) return x to the power y.
SQRT  (x) return the square root of x.
FABS  (x) return the absolute value of x.
FLOOR (X) return the next lowest integer to x.
CEIL  (x) return the next highest integer to x.
ATOF  (str) convert string to double floating point.
ATOC  (str) convert string to integer.
ATOL  (str) convert string to long integer.
FREXP (x, exp, ptr) }
LDEXP (x, exp)      } convert floating point numbers to exponent
MODF  (x, int, tr)  } and mantissa.
SIN   (x) return sine of x.
COS   (x) return cosine of x.
TAN   (x) return tangent of x.
ASIN  (x) return arcsine of x.
ACOS  (x) return arccosine of x.
ATAN  (x) return arctangent of x.
ATAN2 (x, y) return arctangent of x/y.
HYPOT (x, y) return sqrt(x^2 + y^2).
SINH  (x) return hyperbolic sine of x.
COSH  (x) return hyperbolic cosine of x.
TANH  (x) return hyperbolic tangent of x.
```

2.3.6 Renamed MUSS Functions

The following procedures have been added to MUNX because of name clashes with MUSS system procedures. These procedures merely call the MUSS system procedures.

```
MUSSWAIT   -- MUSS WAIT function
MUSSKILL   -- MUSS KILL function
MUSSSTRING -- function to convert 'C' type string (terminated
    with an ASCII NUL character) to a MUSS type bounded string.
```

2.3.7 High level I/O Procedures

The high level I/O functions use buffered I/O and the data structures and declarations needed to support this are in the header file "MU6X:HDR/stdio.h". These define several macros in common use, in particular for single character I/O, and the type FILE, which is a structure used to identify the file in use. Pointers to type FILE are returned by fopen and its relatives, and used in other file operations. The constant file pointers stdin, stdout and stderr are also defined and are used to access the standard input, output and error files. This file should be included into any program using the standard I/O package with the line

#include "MU6X:HDR/stdio.h"

These procedures are part of the standard I/O library.

```
REALLOC  (ptr, newsize)
FREE     (ptr)
MALLOC   (size)
CFREE    (ptr, num, sizeofname)
CALLOC   (num, sizeofname)
CLEARERR (fp)
XXEXlT   (status)
FDOPEN   (fp, mode)
FGETC    (fp)
FGETS    (line, max, fp)
FCLOSE   (fp)
FFLUSH   (fp)
FOPEN    (name, mode)
FPRINTF  (fp, format, arg ...)
FPUTC    (char, fp)
FPUTS    (line, fp)
FREAD    (buffer, size, number, fp)
FREOPEN  (name, mode, fp)
FSEEK    (fp, offset, direction)
FTELL    (fp)
FWRITE   (buffer, size, number, fp)
GETS     (buff)
GETW     (fp)
PRINTF   (format, args ...)
PUTCHAR  (char)
PUTS     (str)
```

```
PUTW     (fp, word)
SSCANF   (string, format, args ...)
FSCANF   (fp, format, args ...)
SCANF    (format, args ...)
SETBUF   (adr, fp)
SPRINTF  (str, format, args ...)
UNGETC   (char, fp).
```

2.3.8 Macro Procedures

```
FEOF     (fp)
FERROR   (fp)
CLEARERR (fp)
FILENO   (fp).
```

2.3.9 Extensions

3 THE CORE GRAPHICS PACKAGE

3.1 CORE GRAPHICS FACILITIES

3.1.1 Introduction

This section outlines the graphics facilities provided in MUSS which functionally conform to the GSPC CORE System Graphics Standard (It is abbreviated as "CORE" in the latter description). A complete description of the CORE standard may be found in *Computer Graphics* 13 (3) Aug. 79 *Status Report of the Graphics Standard Planning Committee*. For the benefit of users unfamiliar with the CORE Standard, there follows a brief outline of its concepts and its interface with MUSS.

The CORE communicates with devices through a KERNEL which contains a device independent part for interpreting commands from CORE and translating them into calls on any of several separate device drivers. Device types currently supported are:

1) Genisco display.
2) Benson Plotter.
3) MC68000 memory mapped display.
4) Hewlett Packard Plotter.
5) VIP Raster Display.
6) VME Raster Display.

The facility exists for driving workstations containing only a KERNEL from a CORE/KERNEL running on another machine in the network. This is achieved by providing a special device type in the KERNEL which is driven in the same way as other devices. However, for this device the KERNEL merely passes commands as MUSS messages to the KERNEL in the workstation where it finally appears on a pre-selected device.

3.1.2 How to access CORE

The CORE system is available as two libraries UTIL:CORE and UTIL:KERNEL. The CORE is a Fortran library with the restriction that only six character names are used. This means that all CORE standard procedure names are called by a six character mnemonic. Each of these appears alongside its equivalent core name in a description of the available command set in section 3.1.4.

3.1.3 Core System Overview

The CORE provides a suite of procedures for producing graphical output and receiving interactive input. Facilities supported fall into six categories as follows:

1 output primitives
2 viewing
3 picture segmentation and naming
4 attributes
5 control
6 input primitives

The version implemented has the following specification:

Output level 3C	– dynamic output with full image transformation
Input level 1	– no input
Dimension level 2	– three dimensional
Hiddensurf level 1	– no hidden surface removal

In CORE, the user constructs a picture by invoking "output primitives", placed in a coordinate system called "world coordinate space". This is user-defined, and may be two or three-dimensional, expressed in arbitrary units which the user might regard as metres, miles, inches, etc. This space must be eventually mapped to the physical display device, or "view surface", and this mapping comes under the user's control and is called the "viewing operation".

The CORE must be initialised by first issuing INITIALISE.CORE command. Also before a view surface can be used, it must be INITIALISED and then SELECTED. When it is no longer required, it is DESELECTED and TERMINATED. More than one device may be initialised at any one time. An INITIALISED but DESELECTED device may be SELECTED again providing the current one is first DESELECTED.

Thus a typical sequence of commands to use the CORE for a general user is shown below. These are expressed as Fortran calls on the six character mnemonic names in the CORE library. Their core equivalent is shown on the right.

```
INIT(OUTER.LEV INNER.LEV, DIM          ::  INITIALISE CORE
DEF.DEV(LOGICAL.DEV, DESTINATION, DEV.TYPE, OUTPUT.MODE)
```

```
NITSRF(LOGICAL.DEV)                    ::  INITIALISE VIEWSURFACE
SELSRF(LOGICAL.DEV)                    ::  SELECT VIEWSURFACE
SWINDO(XMIN, XMAX, YMIN, YMAX)         ::  SET WINDOW
SNDCS2 SET.NDCSPACE.2(XMAX, YMAX)      ::  SET NDC.SPACE2
SVPRT2 SET.VIEWPORT.2(XMIN, XMAX, YMIN, YMAX)::  SET.VIEWPORT2
          1           (Attribute or
          1              Output commands)
          1
DELSRF (LOGICAL.DEV)                   ::  DESELECT VIEWSURFACE
TRMSRF (LOGICAL.DEV)                   ::  TERMINATE VIEWSURFACE
TERM()                                 ::  TERMINATE CORE
```

Here the command DEF.DEV is an additional command which is available through the KERNEL library for creating a correspondence between a logical device number of the CORE and an actual device on the MUSS system.

Other commands exist in the KERNEL for the convenience of typical user requirements. Some of these merely combine the effect of several of those described above and others are for more specific user requirements. For example a user who is producing 2D drawings on A4 paper may use a sequence of commands such as:

```
INIT.CORE.STR (DESTINATION, DEV.TYPE)
A4.FRAME (WORLD.X.MAX, WORLD.Y.MAX)
             1
             1
             1
             1
             1
        TERM.STR ()
```

Output Primitives

The CORE provides four primitives for graphical output, as follows

```
line
text
marker
polygon
```

The CORE maintains a conceptual "current drawing position" (CP) measured in the world coordinate system. CP is changed when some output

commands, e.g. drawing line, marker, etc. are executed. The appearance of the output, produced by the output primitives is determined by global "output primitive attributes", each of which may take one of a number of standard values. The attributes are:

```
line style
line width
colour
intensity
character string attributes
marker symbol
pen
```

The Viewing Operation

It is convenient to consider two dimensional (2D) and three-dimensional (3D) viewing separately.

The user places output primitives in a 2D world coordinate system. To specify what part of the complete picture is to be displayed, a rectangular "window" over the world coordinate space may be defined. Only primitives appearing within this window are considered for mapping onto the view surface. To specify where on the view surface the contents of the window are to be displayed, the user defines a "viewport" on the view surface.

3D viewing is inherently more complex than the 2D case due to the fact that the user is specifying 3D objects, and the display device is only 2D. This solution to this mismatch is to use a "projection", which transforms a 3D object onto a 2D projection plane.

Corresponding to the 2D viewing window, is a 3D view volume. Only primitives appearing within this volume are considered further. The user next specifies a projection, which may be parallel or perspective, onto a projection plane. Next a window over the projection plane may be specified, and contents of this window are mapped to a viewport specified over the view surface.

The foregoing description of the viewing operation is intended as a brief synopsis. For a more complete specification relating to the CORE commands described later the user is referred to the CORE Standard.

Picture Segmentation and Naming

The user introduces structure into the picture by grouping together output

primitives into named picture segments, called "retained segments". Retained segments have attributes, as follows:

```
visibility
high lighting
detectability
image transformation.
```

The Image of a retained segment may be scaled, shifted and rotated using an "Image transformation".

Attributes

Commands are provided to enquire and modify the current values of all the attributes mentioned above.

Control

The user has control over the selection of surfaces, and the changing of displayed pictures. CORE system errors are reported to the user, with an indication of their severity.

Input Primitives

Not implemented.

3.1.4 Commands

These are categorised in 6 classes:

1. CONTROL COMMANDS
2. GRAPHICAL OUTPUT COMMANDS
3. VIEWING COMMANDS
4. ATTRIBUTE COMMANDS
5. PICTURE SEGMENTATION AND NAMING COMMANDS
6. PIXEL COMMANDS

Control Commands

The four control commands available through the KERNEL are described first.

```
1) DEF.DEV (INTEGER, ADDR [LOGICAL8], LOGICAL64, INTEGER) INTEGER
```

This command creates a correspondence between a viewsurface number and a specific physical device. P1 specifies the viewsurface number which is defined by the user. P2 specifies the destination for the output in the case of devices which are connected as terminals or when device type is a memory mapped device it specifies a particular device while more than one device of the same device type are available. P3 specifies the device type. Five device types which have been defined are:

```
KER        denotes the virtual device KERNEL
GEN        denotes the raster color device GENISCO
BEN        denotes the BENSON plotter
MOT        denotes the memory mapped MOTOROLA monitor
HPP        denotes the HEWLETT PACKARD plotter
VME        denotes the VME monitor
VIP        denotes the VIP raster colour device
MT1        denotes the memory mapped MOTOROLA monitor
               with hidden surface compatibility
```

P4 specifies the 'mode' in the case where PS specifies an output destination. There are three modes

```
1          Break output every command
2          Break output every frame or segment
3          Only break output when the viewsurface is terminated.
```

Mode 1 causes the output to use short messages and the other modes cause it to use long messages.

Devices which are driven by a batch or online output stream (i.e. pen plotters and some colour displays) have the stream number monitored as the result. This stream is automatically selected/deselected by the SELECT.VIEW.SURFACE/DESELECT.VIEW.SURFACE commands and so is only made available for stream switching for monitoring purposes. Devices not requiring an output stream have a positive value for the result. A negative result is set to indicate command failure.

`2) INIT.CORE.STR (ADDR [LOGICAL8], LOGICAL8, INTEGER) INTEGER`

This command calls DEF.DEV to define a viewsurface as the destination for the graphics output of the type specified by P2. It initialises the CORE and the viewsurface, and selects the viewsurface.

P1 specifies the destination to which the graphics output is sent.

P2 specifies the device type.

P3 specifies a mode for the output stream set up as the destination as in DEF.DEV.

The result is the output stream number (if required) and is given only fer monitoring purposes. A negative result indicates command failures.

```
3) TERM.STR ( )
```

This command deselects the selected viewsurface, terminates the viewsurface and terminates the CORE.

```
4) A4.FRAME (INTEGER, INTEGER)
```

This command sets the mapping from the world coordinate space on to the physical coordinate space of A4 page size.

P1, P2 specify the XMAX and YMAX respectively in the world coordinate system, i.e. the window is set to (0.0, P1, 0.0, P2).

```
CORE Commands
```

The CORE control commands are now listed with their six character mnemonic name alongside. Descriptions are given for some of the more frequently used commands otherwise the user must refer to the CORE standard description.

```
1) INITIALISE.CORE (INTEGER, INTEGER, INTEGER)        INIT
```

This command initialises the CORE and sets all default parameters, P1 through P3 respectively specify Output, Input and Dimension. A suitable choice would be:

```
P1 = 3
P2 = 0
P3 = 2
```

```
2) TERMINATE.CORE ()                                  TERM
```

This command terminates the CORE and releases all graphics devices.

```
3) INITIALISE.VIEW.SURFACE (INTEGER)                  NITSRF
```

This command initialises the view surface number specified by P1 ready for use by the CORE.

```
4) TERMINATE.VIEW.SURFACE (INTEGER)                   TRMSRF
```

This command terminates the view surface specified by P1.

```
5) SELECT.VIEW.SURFACE (INTEGER)                    SELSRF
```

This command selects the view surface number specified by P1 for use.

For devices requiring an output stream, it is selected here and is assumed to remain selected until DESELECT.VIEW.SURFACE is called.

```
6) DESELECT.VIEW.SURFACE (INTEGER)                  DELSRF
```

This command deselects the view surface by P1, and selects the output in use prior to using CORE.

```
7) NEW.FRAME ()                                     NEWFRM
```

This command causes a 'new frame' action on the currently selected view surface. In the case of CRT devices the screen is cleared. For a plotter or printer device, the drawing surface is moved on to a clean area.

Other commands available:

```
SET.IMMEDIATE.VISIBILITY (IMMEDIACY)          SIMVIS
BEGIN.BATCH.OF.UPDATES ( )                    BGNUPD
END.BATCH.OF.UPDATES ( )                      ENDUPD
INQUIRE.CONTROL.STATUS (IMMEDIACY.BATCHING)   ICONST
```

Graphical Output Primitive Commands

Most of the graphical output PRIMITIVE commands affect the current position (CP), a user – maintained value that defines the current drawing position in WORLD COORDINATES. The CP is used to determine the starting point for LINE.POLYLINE and TEXT. The CP is initialised to (O,O) when the CORE is initialised.

```
1) MOVE.ABS.2 (REAL, REAL)                          MOVA2
```

This command sets CP to the position (x,y) as specified by P1 and P2 respectively.

```
2) MOVE.REL.2 (REAL, REAL)                          MOVR2
```

This command is used to increment the CP by (x,y) specified by P1 and P2 respectively.

```
3) LINE.ABS.2 (REAL, REAL)                          LINA2
```

This command a line drawn from CP to the (x,y) position specified by P1 and P2 respectively. The final position becomes the new CP.

```
4) LINE.REL.2 (REAL, REAL)                                    LINR2
```

This command describes a line from CP to (CP+x, CP+y). where x and y are specified by P1 and P2 respectively. The final position becomes the new CP.

```
5) POLYLINE.ABS.2 (ADDR [REAL], ADDR [REAL], INTEGER)  PUNA2
```

This command describes a connected sequence of lines, P1 and P2 respectively specify arrays of x and y coordinates, each of length n specified by P3. The sequence starts at CP and runs next to (x.array [1], y.array [1]) and ends at (x.array [n], y.array [n]). The final position becomes the new CP.

```
6) POLYLINE.REL.2 (ADDR [REAL], ADDR [REAL], INTEGER)  PLINR2
```

This command is similar to POLYLINE.ABS.2 except that P1 and P2 respectively specify arrays of x and y relative coordinates, each of length n specified by P3.

```
7) MARKER.ABS.2 (REAL, REAL)                                  MARKA2
```

This command sets CP to the point (x,y) given by P1 and P2 respectively, and displays the current marker symbol.

```
8) MARKER REL.2 (REAL, REAL)                                  MARKR2
```

This command is similar to MARKER.ABS.2 except that the CP is incremented by (x,y) as specified by P1 and P2 respectively before the marker is drawn.

```
9) POLYMARKER.ABS.2 (ADDR [REAL], ADDR [REAL], INTEGER) PMRKA2
```

This command describes a sequence of marker symbols to be drawn. As POLYLINE, P1 and P2 respectively specify arrays of x and y coordinates, each of length n given by P3. The current marker is drawn at (x.array [1], y.array [1]) through (x.array [n], y.array [n]). The CP is updated to (x.array [n], y.array [n]).

```
10) POLYMARKER.REL.2 (ADDR [REAL], ADDR [REAL], INTEGER) PMRKR2
```

This command is similar to POLYMARKER.ABS.2 except that P1 and P2 respectively specify arrays of x and y relative coordinates, each of length n given by P3.

```
11) TEXT (ADDR [LOGICAL8], INTEGER)                    TEXTT
```

This command displays the text string specified by P1, of length given by P2, at CP, which remains unaffected by the command. Spaces within the string to be drawn should be indicated in an interactive mode by !20!.

Other commands available:

```
POLYGON.ABS.2 (X.ARRAY,Y.ARRAY,N)                      POLYA2
POLYGON.ABS.3 (X.ARRAY,Y.ARRAY,Z.ARRAY,N)              POLYA3
POLYGON.REL.2 (DX.ARRAY,DY.ARRAY,N)                    POLYR2
POLYGON.REL.3 (DX.ARRAY,DY.ARRAY,DZ.ARRAY,N)           POLYR3
MOVE.ABS.3 (X,Y,Z)                                     MOVA3
LINE.REL.3 (DX,DY,DZ)                                  MOVR3
LINE.ABS.3 (X,Y,Z)                                     LINA3
LINE.REL.3 (DX,DY,DZ)                                  LINR3
POLYLINE.ABS.3 (X.ARRAY,Y.ARRAY,Z.ARRAY,N)             PLINA3
POLYLINE.REL.3 (DX.ARRAY,DY.ARRAY,DZ.ARRAY,N)          PLINR3
MARKER.ABS.3 (X,Y,Z)                                   MARKA3
MARKER.REL.3 (DX,DY,DZ)                                MARKR3
POLYMARKER.ABS.3 (X.ARRAY,Y.ARRAY,Z.ARRAY,N)           PMRKA3
POLYMARKER.REL.3 (DX.ARRAY,DY.ARRAY,DZ.ARRAY,N)        PMRKR3
INQUIRE.CURRENT.POSITION.2 (X,Y)                       IPSN2
INQUIRE.CURRENT.POSITION.3 (X,Y,Z)                     IPSN3
```

Viewing Commands

The following commands are used to control the viewing operation.

```
1) SET.WINDOW (REAL, REAL, REAL, REAL)                 SWINDO
```

This command allows the user to specify a rectangular window over his world coordinate space. P1 through P4 specify XMIN, XMAX, YMIN and YMAX of the sides of the rectangle respectively in world coordinates. The default window is (O,1,0,1).

```
2) SET.WINDOW.CLIPPING (INTEGER)                       SCLIPW
```

This command controls window clipping. P1 = 0 sets clipping off and P1 = 1 sets clipping on. Window clipping is on by default.

```
3) SET.NDC.SPACE.2 (REAL, REAL)                        SNDCS2
```

This command defines the size of the 2D normalised device coordinate space for all view surfaces used by the CORE. P1 and P2 specify width

and height respectively, and must both lie in the range 0 to 1. At least one parameter must have a value of 1. The default NDC space has width=1 and height=1 and is mapped onto a square area of the view surface. By defining a rectangular NDC space it is possible to utilise the entire view surface.

Specifically

```
GENISCO              : x = 1.0, y = 0.8
MOTOROLA             : x = 1.0, y = 0.73
HEWLETI PACKARD      : x = 0.722, y = 1.0
VME                  : x = 1.0, y = 0.75
```

If this command is not issued, default NDC space is assumed.

```
4) SET.VIEWPORT.2 (REAL, REAL, REAL, REAL)          SVPRT2
```

This command defines the extent of the current viewport rectangle. P1 through P4 specify XMIN, XMAX, YMIN, YMAX of the rectangle respectively, expressed in normalised device coordinates. The default is the entire NDC space.

```
5) MAP.NDC.TO.WORLD.2 (REAL, REAL, ADDR REAL, ADDR REAL) MNTOW2
```

This command is used to determine the world coordinate position corresponding to a NDC position. P1 and P2 respectively define the point (x,y) in NDC space. The corresponding position in world coordinates, (x1,y1) is returned in P3 and P4.

```
6) MAP.WORLD.TO.NDC.2 (REAL, REAL, ADDR REAL, ADDR REAL) MWTON2
```

This command does the inverse of the previous command. The world position (x,y) defined by P1 and P2 is mapped to the NDC position (x1,y1) which is returned in P3 and P4.

The following enquiry commands are supported. In each command, the values of the specified viewing parameters are copied into the specified parameters.

```
7) INQUIRE.WINDOW (ADDR REAL, ADDR REAL, ADDR REAL,
      ADDR REAL                                      IWINDO
```

The parameters are:

```
      P1 XMIN
```

```
        P2 XMAX
        P3 YMIN
        P4 YMAX

8) INQUIRE.VIEWING.CONTROL.PARAMETERS (ADDR INTEGER,
ADDR INTEGER, ADDR INTEGER, ADDR INTEGER)          IVCPAR
```

The parameters are:

```
        P1 WINDOW CUPPING: 0 = off, 1 = on
        P2 FRONT CUPPING: 0 = off, 1 = on
        P3 BACK CUPPING: 0 = off, 1 = on
        P4 TYPE

9) INQ.NDC.SPACE.2 (ADDR REAL ADDR REAL)           INDCS2
```

The parameters are:

```
        P1 width
        P2 height

10) INQ.VIEWPORT.2 (ADDR REAL, ADDR REAL, ADOR REAL,
ADDR REAL)                                         IVPRT2
```

The parameters are:

```
        P1 XMIN
        P2 XMAX
        P3 YMIN
        P4 YMAX
```

Other commands available:

```
 SET.VIEW.UP.2 (DX.UP, DY.UP)                       SVUP2
 INQUIRE.VIEW.OP.2 (DX.UP,DY.UP)                    IVUP2
 SET.WORLD.COORDINATE.MATRIX.2 (MATRIX.2)          SMWTX2
 INQUIRE.WORLD.COORDINATE.MATRIX.2 (MATRIX.2)      IWMTX2
 SET.VIEW.REFERENCE.POINT (X.REF,Y.REF,Z.REF)      SVRFPT
 SET.VIEW.PLANE.NORMAL (DX.NORM,DY.NORM,DZ.NORM)   SVPNOR
 SET.VIEW.PLANE.DISTANCE (VIEW.DISTANCE)           SVPDIS
 SET.VIEW.DEPTH (FRONT.DISTANCE,BACK.DISTANCE)     SVDPTH
 SET.PROJECTION                                     SPROJ
          (PROJECTION.TYPE, DX.PROJ, DY.PROJ, DZ.PROJ)
```

```
SET.VIEW.UP.3 (DX.UP, DY.UP, ,DZ.UP)                SVUP3
SET.NDC.SPACE.3 (WIDTH,HEIGHT,DEPTH)               SNDCS3
SET.VIEWPORT.3 (XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX)     SVPRT3
SET.VIEWING.PARAMETERS (VIEWING.PARAMETER.ARRAY)   SVPARM
INQUIRE.VIEW.REFERENCE.POINT (X.REF,Y.REF,Z.REF )  IVRFPT
INQUIRE.VIEW.PLANE.NORMAL                          IVPNOR
         (DX.NORM,DY.NORM,DZ.NORM)
INQUIRE.VIEW.PLANE.DISTANCE (VIEW.DISTANCE)        IVPDIS
INQUIRE.VIEW.DEPTH (FRONT.DISTANCE,BACK.DISTANCE   IVDPTH
INQUIRE.PROJECTION                                  IPROJ
(PROJECTION.TYPE, DX.PROJ, DY.PROJ, DZ.PROJ)
INQUIRE.VIEW.UP.3 (DX.UP, DY.UP, DZ.UP)             IVUP3
INQUIRE.MDC.SPACE.3 (WIDTH, HEIGHT, DEPTH)         INOCS3
INQUIRE.VIEWPOINT.3                                IVPRT3
      (XMIN, XMAX,YMIN,YMAX,ZMIN,ZMAX)
INQUIRE.VIEWING.PARAMETERS                         IVPARM
       (VIEWING.PARAMETER.ARRAY)
MAP.NDC.TO.WORLD.3 (NDC.X,NDC.Y,NDC.Z,X,Y,Z)       MNTOW3
MAP.WORLD.TO.NDC.3 (X,Y,Z,NDC.X,NDC.Y,NDC.Z)       MNTON3
SET.FRONT.PLANE.CLIPPING (ON OFF)                  SCLPFP
SET.BACK.PLANE.CLIPPING (ON OFF)                   SCLPBP
SET.COORDINATE.SYSTEM.TYPE (TYPE)                  SCORTP
SET.WORLD.COORDINATE.MATRIX.3 (MATRIX.3)           SWMTX3
INQUIRE.WORLD.COORDINATE.MATRIX.3 (MATRIX.3)       IWMTX3
```

Control of Colour

In the CORE System, colour is specified by an *index*. The relationship between index and colour is device dependent. In some devices the correspondence is fixed and in others it may be defined by the command

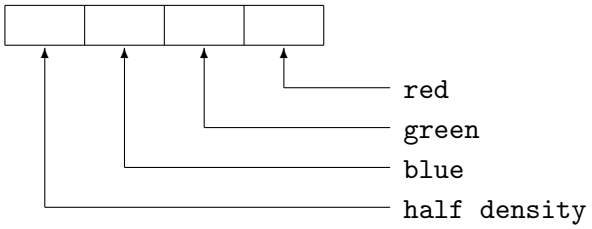
```
1) DEFINE.COLOR.INDEX (INTEGER,INTEGER,INTEGER,INTEGER) DCLNDX
```

This defines the colour on the view surface P1 associated with index P2. The parameters P3, P4, PS define the intensity of red, blue and green in the specified colour. The range of values for the index and the individual primary colour components is device dependent.

There are four devices in use on the Manchester University systems which involve colour

a) Motorola bit mapped displays – on these devices the colour index is in

the range 0-15 and the associated colours are fixed. The 4-bit index is bit significant thus



b) The VIP display – this has a 4-bit colour index. Each of the values 1-15 may be defined as a combination of 0-15 intensity levels of the three primary colours. It is initialised to have the same colour range as (a).

c) The Genisco – this has a 8-bit colour index. Each of the 255 values 1-255 may be associated with a combination of 0-15 intensity levels of the primary colours. The values 0-15 are initialised to the same colours as (a). Each subsequent group of 16 is similarly initialised but with the full intensity level reduced by unit each time.

d) Benson – this pen plotter has 3 pens and the index selects a pen.

e) The VME display – this has a fixed grey scale in the range 0 – 7. 0 is black and 7 full intensity green.

Other commands available:

```
INQUIRE.COLOR.INDEX (SURF,INDEX,C1,C2,C3)        ICLNDX
DEFINE.COLOR.INDICES                             DCLNDC
(SURF,I1,I2,C1ARRAY,C2ARRAY,C3ARRAY)
INQUIRE.COLOR.INDICES                            ICLNDC
(SURF,I1,I2,C1ARRAY,C2ARRAY,C3ARRAY)
DEFINE.STANDARD.COLOR.INDICES                    DSTDCL
(SURF,L1,L2,L3,LOW1,LOW2,LOW3,HIGH1,HIGH2,HIGH3)
INQUIRE.STANDARD.COLOR.INDICES                   ISTDCL
(SURF,L1,L2,L3,LOW1,LOW2,LOW3,HIGH1,HIGH2,HIGH3)
DEFINE.INTENSITY.INDEX (SURF,I,INTENSITY)        DINNDX
INQUIRE.INTENSITY.INDEX (SURF,I,INTENSITY)       IINNDX
DEFINE.INTENSITY.INDICES (SURF,I1,I2,ARRAY)      DINNOC
INQUIRE.INTENSITY.INDICES (SURF,I1,I2,ARRAY)     IINNDC
SET.COLOR.MODEL (MODEL)                          SCLMOD
```

```
INQUIRE.COLOR.MODEL (MODEL)                    ICLMOD
SET.BACKGROUND.INDEX (INDEX)                   SBGNDX
INQUIRE.BACKGROUND.INDEX (INDEX)               IBGNDX
```

Attribute Commands

1) SET.LINE.INDEX (INTEGER) SLNDX

This command selects the colour to be used when drawing lines or markers. P1 specifies the colour index. The default index is 1.

2) SET.TEXT.INDEX (INTEGER) STNDX

This command defines the colour to be used when drawing text. P1 specifies the colour index. The default index is 1.

3) SET.PEN (INTEGER) SPEN

This command applies only to the Benson plotter and is used to select one of three pens (P1 = 0, 1 or 2). Pen 0 is selected by default.

4) INQ.LINE.INDEX (ADDR INTEGER) ILNDX

This command returns the current value of the line index in P1.

5) INQ.TEXT.INDEX (ADDR INTEGER) ITNDX

This command returns the current value of the text index in P1.

6) INQ.PEN (ADDR INTEGER)

This command returns the number of the currently selected pen in P1.

7) SET.MARKER.SYMBOL (INTEGER) SMKSYM

This command enables the user to select a marker symbol from the standard symbol table to become the current marker symbol. P1 specifies the index into this table. The Graphics System marker table contains the following markers:

Table Index	Marker Symbol
1	.
2	+
3	*
4	0
5	X
6	>
7	-
8	↑
9	#

The Graphics System default marker symbol is ".".

8) INQ.MARKER.SYMBOL (ADDR INTEGER) IMKSYM

This command returns the index value of the currently selected marker symbol in P1.

9) INQ.TEXT.EXTENT.2 (ADDR [LOGICAL8], INTEGER, ADDR REAL, ADDR REAL)

This command is used to discover the extent of the character string specified by P1. If drawn at the CP on view surface P2. The extent in the x direction is returned in P3, and in the y direction in P4.

10) SET.LINE.STYLE (INTEGER) SLSTYL

This command enables the user to specify in which style lines and polygons are drawn. P1=1 specifies solid lines. P2=2 specifies dashed lines.

Solid lines are assumed by default.

11) INQ.LINE.STYLE (ADDR INTEGER) ILSTYL

This command returns the value of the currently selected linestyle in P1. See SET.LINESTYLE for a description of the "linestyle" attribute.

12) SET.CHAR SIZE (REAL.REAL) SCHSIZ

This command specifies the character size to be used when drawing text. P1 gives the width and P2 the height of each character in world coordinates. This command has effect only on the Benson and Hewlett Packard plotters. The Genisco and Motorola displays do not have sufficient screen resolution.

13) SET.CHAR.PATH CINTEGER) SCHPTH

This command specifies the direction in which text strings are to be drawn. Meaningful values of P1 are as follows :

```
P1 = 0 :  right.  This is the default.
P1 = 1 :  up
P1 = 2 :  left
P1 = 3 :  down
```

This command is only meaningful for the Gensico, Benson and Hewlett Packard plotters.

14) INQ.CHAR.SIZE (ADDR REAL.ADDR REAL ICHSIZ

The height of a character in world coordinates is returned in P1, the width in P2.

15) INQ.CHAR.PATH (ADDA INTEGER) ICHPTH

The currently set character path attribute is copied into P1.

See SET.CHAR.PATH for a description of meaningful values.

Other commands available:

```
SET.POLYGON.INTERIOR.STYLE (STYLE)                SPISTY
SET.POLYGON.EDGE.STYLE (STYLE)                    SPESTY
SET.FILL.INDEX (INDEX)                            SFNDX
SET.VERTEX.INDICES (VERTEX.ARRAY,N)               SVINDC
SET.VERTEX.COLORS (ARRAY,IN)                      SVCOLS
SET.VERTEX.INIENSITIES (ARRAY,N)                  SVINTS
SET.LINE.COLOR (COL1,COL2,COL3)                   SLNCOL
SET.FILL.COLOR (COL1,COL2,COL3)                   SFLCOL
SET.TEXT.COLOR (COL1,COL2,COL3)                   STXCOL
SET.LINE.INTENSITY (INTENSITY)                    SLNINT
SET.FILL.INTENSITY (INTENSITY)                    SFLINT
SET.TEXT.INTENSITY (INTENSITY)                    STXINT
INQUIRE.POLYGON,INTERIOR.STYLE) (STYLE)           IPISTY
INQUIRE.POLYGON.EDGE.STYLE (STYLE)                IPESTY
INQUIRE.FILL.INDEX (INDEX)                        IFNDX
INQUIRE.VERTEX.INDICES (N1,ARRAY,N2)              IVINDC
INQUIRE.VERTEX.COLORS (N1,ARRAY,N2 )              IVCOLS
INQUIRE.LINE.INTENSITIES (N1,ARRAY,N2)            IVINTS
INQUIRE.LINE.COLOR (COL1,COL2,COL3)               ILNCOL
INQUIRE.FILL.COLOR (COL1,COL2,COL3)               IFLCOL
INQUIRE.TEXT.COLOR (COL1,COL2,COL3)               IFXCOL
INQUIRE.LINE.INTENSITY (INTENSITY)                ILNINT
```

INQUIRE.FILL.INTENSITY (INTENSITY)	IFLINT
INQUIRE.TEXT.INTENSITY (INTENSITY)	ITXINT
SET.CHARPLANE (DX.PLANE,DY.PLANE,DZ.PLANE)	SCHPLA
SET.CHARUP.3 (DX,CHARUP,DY,CHARUP,DZ,CHARUP)	SCHPU3
SET.LINEWIDTH (LINEWIDIH)	SLWID
SET.FONT (FONT)	SFONT
SET.CHARUP.2 (CHARUP,DY,CHARUP)	SCHUP2
SET.CHARSPACE (CHARSPACE)	SCHSPA
SET.CHARJUST (CHARJUST)	SCHJST
SET.CHARPRECISION (CHARPRECISION)	SCHPRE
SET.PICK.ID (ID)	SPID
SET.PRIMITIVE.ATTRIBUTES.2 (PRIMITIVE.ATTRIBUTES.ARRAY.2)	SPRAT2
INQUIRE.LINEWIDTH (LINEWIDTH)	ILWID
INQUIRE.FONT (FONT)	IFONT
INQUIRE CHARUP.2 (DX.CHARUP,DY.CHARUP)	ICHUP2
INQUIRE.CHARSPACE (CHARSPACE)	ICHSPA
INQUIRE.CHARJUST (CHARJUST)	ICHJST
INQUIRE.CHARPRECISION (CHARPRECISION)	ICHPRE
INQUIRE.PICX.ID (ID)	IPID
INQUIRE.PRIMITIVE.ATTRIBUTES.2 (PRIMITIVE.ATTRIBUTE.ARRAY.2)	IPRAT2
SET.PRIMITIVE.ATTRIBUTES.3 (PRIMITIVE.ATTRIBUTE.ARRAY.3)	SPRAT3
INQUIRE.CHARPLANE (DX.PLANE,DY.PLANE,DZ.PLANE)	ICHPLA
INQUIRE.CHARUP.3 (DX.CHARUP,DY.CHARUP,DZ.CBARUP)	ICHUP3
INQUIRE.PRIMITIVE.ATTRIBUTES.3 (PRIMITIVE.ATTRIBUTE.ARRAY.3)	IPRAT3
SET.VISIBILITY (VISIBILITY)	SVISIB
SET.HIGHLIGHTING (HIGHLIGHTING)	SHILIT
INQUIRE.VISIBILITY (VISIBILITY)	IVISIB
INQUIRE.HIGHLIGHTING (HIGHLIGHTING)	IHILIT
SET.SEGMENT.VISIBILITY (SEGMENT.NAME.VISIBILITY)	SSVIS
SET.SEGMENT.HIGHLIGHTING (SEGMENT.NAME.HIGHLIGHTING)	SSHILT
INQUIRE.SEGMENT.VISIBILITY (SEGMENT.NAME.VISIBILITY)	ISVIS
INQUIRE.SEGMENT.HIGHLIGHTING (SEGMENT.NAME.HIGHLIGHTING)	ISHILT
SET.DETECTABILITY (DETECTABILITY)	SDTECT

INQUIRE.DETECTABILITY (DETECTABILITY)	IDTECT
SET.SEGMENT, DETECTABILITY	SSDET
(SEGMENT.NAME.DETECTABILITY)	
INQUIRE.SEGMENT.DETECTABILITY	ISDET
(SEGMENT.NAME.DETECTABILITY)	
SET.IMAGE.TRANSFORMATION.TYPE (TYPE)	STRTYP
INQUIRE.IMAGE.TRANSFORMATION.TYPE (TYPE)	ITRTYP
INQUIRE.SEGMENT.IMAGE.TRANSFORMATION.TYPE	ISTTYP
(SEGMENT.NAME.TYPE)	
SET.IMAGE.TRANSLATE.2 (TX,TY)	SITN2
INQUIRE.IMAGE.TRANSLATE.2 (TX,TY)	IITN2
SET.SEGMENT.IMAGE.TRANSLATE.2	SSITN2
(SEGMENT.NAME,TX,TY)	
INQUIRE.SEGMENT.IMAGE.TRANSLATE.2	ISITN2
(SEGMENT.NAME,TX,TY)	
SET.IMAGE.TRANSFORMATION.2 (SX,SY,A,TX,TY)	SITR2
INQUIRE.IMAGE.TRANSFORMATION, 2	IITR2
(SX,SY,A,TX,TY)	
SET.SEGMENT.IMAGE.TRANSFORMATION.2	SSITR2
(SEGMENT.NAME,SX,SY,A,TX,TY)	
INQUIRE.SEGMENT.IMAGE.TRANSFORMATION.2	ISITR2
(SEGMENT.NAME,SX,SY,A,TX,TY)	
SET.IMAGE.TRANSFORMATION.3	SITR3
(SX,SY,SZ,AX,AY,AZ,TX,TY,TZ)	
INQUIRE.IMAGE.TRANSFORMATION.3	IITR3
(SX,SY,SZ,AX,AY,AZ,TX,TY,TZ)	
SET.SEGMENT.IMAGE.TRANSFORMATION.3	SSITR3
(SEGMENT.NAME,SX,SY,SZ,AX,AY,AZ,TX,TY,TZ)	
INQUIRE.SEGMENT.IMAGE.TRANSFORMATION.3	ISITR3
(SEGMENT.NAME,SX,SY,SZ,AX,AY,AZ,TX,TY,TZ)	

Picture Segmentation Commands

1) CREATE.TEMPORARY.SEGMENT ( ) CATSEG

This command creates a new temporary segment into which subsequent output primitives will be placed.

2) CLOSE.TEMPORARY.SEGMENT ( ) CLTSEG

This command closes the currently open temporary segment.

3) INQ.OPEN.TEMPORARY.SEGMENT (ADDR INTEGER) IOTSEG

This command is used to check whether there is currently a temporary segment open. If a segment is open, the command returns the value 1 in P1 otherwise 0.

Other commands available:

```
CREATE.RETAINED.SEGMENT (SEGMENT.NAME)                    CRRSEG
CLOSE.RETAINED.SEGMENT ( ) (3)                            CLRSEG
DELETE.RETAINED.SEGMENT (SEGMENT.NAME)                    DERSEG
DELETE.ALL.RETAINED.SEGMENTS ()                           DELALL
RENAME.RETAINED.SEGMENT (SEGMENT.NAME, ARRAY.SIZE,        RENSEG
VIEW.SURFACE.ARRAY, NUMBER.OF.SURFACES)
INQUIRE.RETAINED.SEGMENT.SURFACES (SEGNAME,NEWNAME)       IRSSRF
INQUIRE.RETAINED.SEGMENT.NAMES                            IRSNAM
(ARRAY.SIZE, SEGMENT.NAME.ARRAY,
NUMBER.OF.SEGMENTS)
INQUIRE.OPEN.RETAINED.SEGMENT (SEGMENT, NAME)             IOSEG
```

Pixel Commands

```
1) SET.PIXEL PATTER.ORIGIN.2 CXABS.YABS)                  SPPOG2
2) INQUIRE.PIXEL.PATTER.ORIGIN.2 CXABS.YABS)              IPPOG2
3) SET.PIXEL.ARRAY (ARRAY.COLS.ROWS)                      SPIXAY
4) INQUIRE.PIXEL.ARRAY (COLS,ROWS,ARRAY,NCOLS,NROWS)      IPIXAY
```

3.2 GKS GPAPHICS

3.3 DIRECT ACCESS GRAPHiCS

Several of the machines supporting MUSS have special graphics hardware which can be accessed directly.

3.3.1 ATV Graphics

The ATV systems have a low resolution colour graphics raster scan monitor which is directly driven through a pixel store. This store can be accessed from user level as common segment 62 and no protection is provided for

simultaneous usage by more than one process as the ATV systems are regarded as single-user workstations.

The monitor has 360 x 264 4-bit pixels addressed consecutively from the start of the common segment with 2 pixels in each byte. The most significant 4 bits of each byte correspond to the left hand pixel of the pair. There is a space at the and of the segment which is used as a pseudo V-store for controlling the effect of the pen and tablet associated with the display. The user can write to some of these pseudo-control words to control scaling between the tablet and the display, to select a display cursor or 'paintspot' option and the colour of the cursor or 'paintspot' displayed. When the cursor option is selected a cursor will be moved around the screen which represents pen position. The paint option allows the user to add or remove colour to or from individual pixels, by means of the pen. The Image created in this way can then be processed or recorded from the pixel store. The remainder of this area is used by the system and should not be written to by the user.

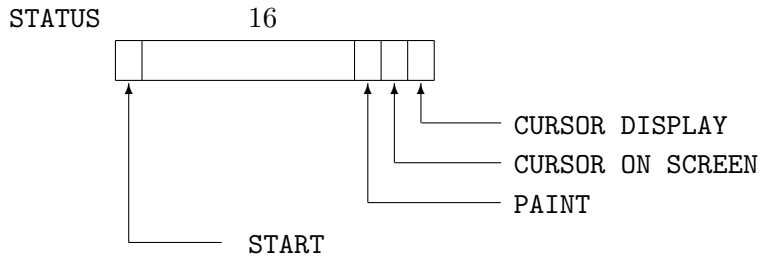
The resolution of the tablet is in general much higher than that of the display so the display is used as a window into the space described by the tablet. The origin of this window on the tablet (the upper left hand corner) can be set in the V-lines X0, Y0 and the viewing area magnified or reduced by scaling factors XS, YS. The current pen position is available X, Y and a STATUS word controls the display at this point on the screen and indicates when the user is required to take some action. When the START bit in the STATUS word is unset, the position of the pen on the surface of the tablet is constantly monitored by the values X and Y and may be displayed on the screen by a cursor in the form of an '+'. The dimensions of the cursor are controlled by the setting of the V-lines CURSOR LENGTH and HEIGHT. And, it may be switched off by resetting the 'cursor display' bit in STATUS. When the the pen (or similar device) next issues a function request, (when the pen is depressed for example) the cursor is removed, the START bit set and no further monitoring occurs until it is reset by the user. This allows the user process to alter the display at X, Y before handing control back to the pen.

Pseudo V-Lines for control of the Tablet

The following table indicates the layout of the V-lines, at addresses in common segment 62.

Address	< – – – – – – 16 – – – – – – >
%FFE0	X
%FFE2	Y
%FFE4	X0
%FFE6	Y0
%FFE8	XS
%FFEA	YS
%FFEC	CURS.LENGTH
%FFEE	CURS.HEIGHT
%FFF0	PEN FUNCTION
%FFF2	STATUS
%FFF4	PCL | PCR
%FFF6	SCL | SCR

X	The X coordinate value of the current pen position on the tablet.
Y	The Y coordinate value of the current pen position on the tablet.
X0	The X coordinate value of the origin (upper left hand corner) of the current display area on the tablet.
Y0	The Y coordinate value of the origin above.
XS	The scaling value in the X-plane required to scale the tablet coordinates down to pixels.
YS	The scaling value in the Y-plane.
CURS.LENGTH	The length of the horizontal cursor line in pixels.
CURS.HEIGHT	The height of the vertical cursor line in pixels.
PCL	paint colour for lefthand pixel.
PCR	paint colour for righthand pixel.
SCL	shade colour for lefthand pixel (i.e. these colour bits are removed).
SCR	shade colour for righthand pixel.
FUNCTION	The number of the function currently issued by the pen/puck /digital handset.



Pen/Tablet Communication interface

When the START bit is reset, characters are received from the pen in the following format:

```
<@> <FUNCTION> <SIGN> <XXXX> <SIGN> <YYYY> <cr>
```

where

<FUNCTION>	is the pen function where 0 – Touching the Tablet surface 1 – Depressed
<SIGN>	is '+' or '-' where '-' values are ignored.
<X> and <Y>	are bytes representing 4 bit hexadecimal numbers such that the number $0 -> 9$ represents the values 0 – 9 and $A -> F$ represent the values 10 – 15.
<cr>	is the carriage return character.

To deal with this interface a mode word is maintained to indicate the current position in the format when a character is received. It has the following significance:

```
0 -- OUT OF SYNC (Waiting for an '@')
1 -- AWAITING FUNCTION
2 -- AWAITING SIGN
3 -- FORMING X
4 -- AWAITING SIGN
5 -- FORMING Y
6 -- AWAITING <cr>.
```

4 TEXT WINDOW FACILITIES

4.1 TEXT WINDOW FACILITIES

4.1.1 Introduction

The facilities provided by this library are concerned with the management and display of several logically distinct windows on character terminals. Two kinds of window are provided. The first is a passive window which functions as an output stream for a process and continuously updates the display. The second is an active window used specifically for the inspection of text and diagrams generated in a textual format. In both cases, but more importantly in the second case, there is the notion of a virtual screen which is in general larger than the window, and may even be larger that the physical screen. An active window may be panned over the virtual screen in order to Inspect the text or diagrams contained on it.

4.1.2 How to Access the Text Window Facilities

The text window facilities exist as a library called WINLIB in the directory of user UTIL. The commands may be accessed by opening this Library with the command:

LIBRARY UTIL : WINLIB

4.1.3 Commands

1) DEFINE.WINDOW (INTEGER, ADDR[LOGICAL], INTEGER32, INTEGER, INTEGER, INTEGER, INTEGER, INTEGER, INTEGER) INTEGER

This procedure defines a passive window. It will normally create two output streams. One of these, the 'window stream' is to be used normally but is set up so that all output on this stream will be intercepted by the window procedures. The second, 'window destination' stream is used only by the window procedures to send output to its ultimate destination.

P1 gives the number of the window to be created. If this is negative, a free window will be selected by the system. P2 and P3 are the same as the second and third parameters of DEFINE.OUTPUT, specifying respectively the ultimate destination of the output and the mode of the window stream. The

location of the window on the physical display surface in terms of character row and column numbers is given by P4 to P7. P4 and P5 respectively give the row and column numbers of the lower left hand corner of the window, and P6 and P7 similarly specify the top right hand corner. If these parameters are zero, the window will have the same dimensions as a normal character display terminal, which is 23 lines of 80 characters. P8 and P9 respectively specify the width and height of the virtual screen in character units. If these parameters are zero, the dimensions of the virtual screen will be implementation – defined. The number of the window stream is returned as the result of the procedure. This stream must be selected in order to send output to the window.

If an existing stream is specified as the destination of the window, it will be used as the window stream.

2) RELEASE.WINDOW (INTEGER)

This procedure is used to release a window which is no longer required. P1 specifies the window stream to be released.

3) VIEW.FILE (ADDR [LOGICAL8], INTEGER, INTEGER, INTEGER, INTEGER, INTEGER, INTEGER)

This procedure is used to define and pan an active window over a virtual screen on which a file is displayed. P1 specifies the file to be inspected. The location of the window on the physical display surface in terms of character row and column numbers is given by P2 to P5. P2 and P3 respectively specify the lower left hand corner of the window, and P4 and P5 similarly specify the top right hand corner. If these parameters are zero, the window will have the same dimensions as a normal character terminal. P6 and P7 specify the width and height of the virtual screen in character units, and if these parameters are zero an installation defined default will be used, which will normally be a width of 240 characters and a height of 125 characters.

Initially the first page of the file is written to the virtual screen and the window is positioned over the top left hand part of the virtual screen. The procedure then prompts for commands to move the position of the window over the virtual screen, and the following commands are recognised:

control P	Display the next page of the file.
←	Shift the window left by one character.
→	Shift the window right by one character.
↑	Shift the window up by one character.
↓	Shift the window down by one character.
control E	Terminate the positioning operation.

Index – Volume 3

A4.FRAME, 23

cc, 8
CLOSE.TEMPORARY.
SEGMENT, 36
cpp, 9
CREATE.TEMPORARY.
SEGMENT, 36

DEF.DEV, 19
DEFINE.COLOR.INDEX, 29
DEFINE.WINDOW, 41
DESELECT.VIEW.SURFACE, 24

ed, 10
EDITORS, 1
EMACS, 1, 10

FRAME, 23

INIT.CORE.STR, 22
INITIALISE.CORE, 23
INITIALISE.VIEW.SURFACE, 23
INQ.CHAR.PATH, 34
INQ.CHAR.SIZE, 34
INQ.LINE.INDEX, 31
INQ.LINE.STYLE, 32
INQ.MARKER.SYMBOL, 32
INQ.NDC.SPACE.2, 28
INQ.OPEN.TEMPORARY.
SEGMENT, 37
INQ.PEN, 31
INQ.TEXT.EXTENT.2, 32
INQ.TEXT.INDEX, 31
INQ.VIEWPORT.2, 28
INQUIRE.PIXEL.
PATTER.ORIGIN.2, 37
INQUIRE.PIXEL.ARRAY, 37
INQUIRE.VIEWING.CONTROL.
PARAMETERS, 28
INQUIRE.WINDOW, 27

LINE.ABS.2, 24
LINE.REL.2, 25

MAP.NDC.TO.WORLD.2, 27
MAP.WORLD.TO.NDC.2, 27
MARKER REL.2, 25
MARKER.ABS.2, 25
MOVE.ABS.2, 24
MOVE.REL.2, 24

NEW.FRAME, 24

pcc, 8
POLYLINE.ABS.2, 25
POLYLINE.REL.2, 25
POLYMARKER.ABS.2, 25
POLYMARKER.REL.2, 25

RELEASE.WINDOW, 42

SELECT.VIEW.SURFACE, 24
SET.CHAR SIZE, 32
SET.CHAR.PATH, 33
SET.LINE.INDEX, 31
SET.LINE.STYLE, 32
SET.MARKER.SYMBOL, 31
SET.NDC.SPACE.2, 26
SET.PEN, 31
SET.PIXEL.ARRAY, 37
SET.PIXEL.PATTER.ORIGIN.2, 37
SET.TEXT.INDEX, 31
SET.VIEWPORT.2, 27
SET.WINDOW, 26
SET.WINDOW.CLIPPING, 26
sh, 10

TERM.STR, 23
TERMINATE.CORE, 23

TERMINATE.VIEW.SURFACE, 23
TEXT, 26

VIEW.FILE, 42